REMARKS

This Amendment is filed in response to the Office action dated June 13, 2003. All objections and rejections are respectfully traversed. Claim one was amended to correct a grammatical error. Claims 9 through 21 were added to better claim the invention. Reconsideration and further examination of the application are respectfully requested.

Applicant submits a formal set of drawings to replace the informal drawings filed with the application. In the formal drawings in Fig. 4, the text associated with reference number 402 was changed from "RECIEVER" to "RECEIVER" and the text associated with reference number 406 was changed from "TOPOLOG" to "TOPOLOGY" to correct misspellings in the originally filed drawings. Applicant respectfully requests that Examiner approve the submitted formal drawings.

In the Office action, Claims 1, 2, 7, and 8 were rejected under 35 U.S.C. §103 as being unpatentable over "Monitoring Distributed Systems" by Joyce et al. (hereinafter "Joyce") in view of U.S. Patent 5,655,081 to Bonnell (hereinafter "Bonnell").

Description of Applicant's Invention

Applicant's invention is directed to a technique for dynamically modifying configuration, settings and other parameters of distributed applications running in a computer network. Clients (e.g., network administrators) associated with the network are automatically notified of changes in the network, such as the addition of an application or process that runs in the network or a new device added to the network. In response to a change, the client may interact with newly opened process, application, device, etc. that

8

triggered the change without having to close or restart a user interface application used by the client to manage the network.

Briefly, upon opening or initializing, an application or process registers with a process manager by issuing a registration service request to the manager. In response, the process manager generates and forwards a notification message containing a reference that identifies the new application/process to a client's user interface application. The client's user interface application, in turn, notifies the client of the new application/process. For new devices, the network includes a topology server that runs a discovery application, which detects the addition of new devices. Similarly, on detecting a new device added to the network, the topology server generates a notification message and forwards the notification message to the client's user interface application. In response to receiving the notification message, the user interface application presents the notification the client. Notably, in accordance with the inventive technique, the user interface application does not have to close and be re-started to present the notification to the client.

Description of the Cited Art

Joyce describes a programming environment (Jade) that supports the development of distributed software. The programming environment includes a distributed monitoring system that supports the observation and control of message passing within a distributed application system consisting of a set of concurrently executing processes. Messages are passed within the Jade environment via an inter-process communications (IPC) facility called "Jipc." See p. 125, Sections 2 and 2.1.

The Jipc facility includes a protocol that enables processes in a Jade distributed system to communicate with each other. Processes exchange messages through the use of a blocking protocol that includes "send" and "reply" primitives. When a Jipc process sends a message to another process, it blocks until it receives a reply. When a reply message is received, the sender is unblocked and allowed to continue executing. See p. 126, Section 2.2.

The distributed monitoring system includes Jipc processes, channel processes, and consoles. A Jipc process is a process that is part of the Jipc system and may be either monitorable or unmonitorable depending on whether it is loaded with a version of Jipc that incorporates monitoring (i.e., a monitorable process is loaded with a version of Jipc that incorporates monitoring). A channel process collects monitoring information from the Jipc processes and distributes this information to one or more consoles. A console examines and interprets the monitoring information and presents it to a user. See pp. 126-127, Section 2.3 and 2.3.1.

Jipc processes may generate monitorable events, which are events that have an effect outside the Jipc process, such as when a Jipc process enters or leaves the Jipc system. When a monitorable event is about to occur in a monitorable Jipc process, information concerning the event is sent to a channel process. The Jipc process is blocked until the channel process replies to the Jipc process at which point the Jipc process continues and the event is allowed to occur. See pp. 128-129, Sections 2.3.1, 2.3.2, and 2.3.3.

Joyce also describes a controller process that enables a user to control the order of events generated by Jipc processes. When an event is about to occur, a monitorable Jipc

process conveys monitoring information to a channel and is blocked. The channel, in turn, forwards the monitoring information to the controller, which, in turn, may postpone replying to the message, thereby suspending operation of the process and preventing the event from occurring. The controller process may continue to receive other monitoring messages from other monitorable Jipc processes that also have an event about to occur. A user can then interact with the controller to control the order of events among the Jipc processes. See p. 129, Section 2.3.2 and p. 142, Sections 4.2 and 4.2.1.

Bonnell describes a method for monitoring and managing applications and other resources in a computer network in a distributed computing environment. According to the method, agent software runs on server systems contained in the network. Each agent carries out various tasks including discovering resources and applications present on a computer system and monitoring particular aspects of the resources and applications. See Col. 6, lines 55-61 and line 67 to Col. 7, lines 1-7.

Consoles (network management computer systems) register with the agents via registration messages that indicate resources, parameters, and events the consoles are interested in receiving information about. In response, the agent loads knowledge modules into its knowledge database as are necessary to provide the monitoring services required by the consoles' registrations. The agent continually monitors the state of these registered resources and sends messages to the consoles registered to receive the information in a manner where a particular console receives only the information it has registered to receive. See Col. 7, lines 15-21 and lines 22-31.

11

Differences Between the Cited Art and Applicant's Invention

Representative claim 1 recites in part:

> "in response to opening the new application or process, *issuing a registration service request from the new application or process to the process manager* through the configuration service layer,"

> "in response to receiving the registration service request at the process manager, *generating and forwarding a notification message that identifies the new application or process to the network management station*" and

> "*automatically displaying the notification message* at the network management station *without having to close and re-start the management station.*"

Applicant respectfully submits that neither Joyce nor Bonnell teach or suggest (either individually or in combination) the steps of *issuing a registration service request from a new application or process to the process manager* nor *generating and forwarding a notification message that identifies the new application or process to the network management station*. Joyce does not describe or even suggest registration service requests. Bonnell does mention a "registration request," however, the request is issued <u>by a console to an agent</u> to inform the agent of information the console wishes to receive from the agent. This is quite different from applicant's registration service *request sent* to a process manager *in response to opening a new application or process* and *generating and forwarding a notification message that identifies the new application or process* to a network management station.

According to Bonnell, an agent process discovers resources and applications on a computer system and monitors particular aspects of the resources and applications present on the system. A console issues a registration request to an agent to inform the agent

as to which resources, parameters and events the console is interested in receiving information about. In response, the agent loads knowledge modules into its knowledge database as are necessary to provide the monitoring services required by the consoles' registrations, monitors the state of the resources and sends messages to the console that contain information the console is interested in receiving. This is quite different from applicant's claimed invention.

Applicant's claimed invention issues *a registration service request to a process manager in response to opening a new application or process*. The *process manager*, in turn, *responds by forwarding a notification message to a network management station*. Applicant's claimed invention does not require a network management station to register with an agent process to be informed of the opening of a new application or process, as is required by Bonnell's teachings. Moreover, unlike Bonnell, registration requests are sent by a new application or process and not by a console.

In addition, the Office action indicates with regards to the system described by Bonnell, "it is inherent that since the display is continuously updated and the applications are continuously monitored that it is not close (sic) and re-started when an update performed."

As noted in the MPEP, the Federal Circuit has ruled that:

> "To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'"

MPEP §2112, quoting <u>In re Robertson</u>, 169 F.3d 743, 745 (Fed. Cir. 1999).

Applicant submits that the Commissioner has not met his burden of demonstrating the consoles in Bonnell provide a continuously updated display representing all resources and application present throughout the network *without having to close and re-start* the consoles.

No extrinsic evidence has been cited in the Office action in support of the contention that Bonnell inherently operates in a manner claimed by the Applicant. Instead having studied Applicant's detailed disclosure of a particular manner of updating a display without having to close and re-start network management station, the Office action simply asserts that, because the display in Bonnell is continually updated and the applications are continuously monitored, that the display is not closed and re-started when an update is performed. Applicant submits that such reasoning cannot support a rejection that is based upon inherency.

In particular, Bonnell fails to make clear as to whether or not the consoles need to be closed and restarted in order to provide a continuously updated display of the resources and application. Rather, Bonnell simply indicates that the console provide a continuously updated display representing the resources and applications throughout the network without further indicating as to whether or not the consoles need to be closed and restarted in order to make this occur.

Applicant respectfully submits that because both Joyce and Bonnell fail to teach applicant's claimed *issuing a registration service request from a new application or process to the process manager, generating and forwarding a notification message that*

identifies the new application or process to network management station and *automatically displaying the notification message* at the network management station *without having to close and re-start the management station*, both Joyce and Bonnell do not render Applicant's claimed invention obvious.

At paragraph 4 of the above-identified Office action, Examiner rejected Claims 3 and 4 as being unpatentable under 35 U.S.C. §103 over Joyce in view of Bonnell and in further view of "Red Hat Linux Unleashed" by Husain (hereinafter "Husain"). Applicant submits that claims 3 and 4 are dependent claims that are dependent on independent claims applicant believes are allowable and therefore should be allowed.

At paragraph 5 of the above-identified Office action, Examiner rejected Claims 5 and 6 as being unpatentable under 35 U.S.C. 103 over Joyce in view of Bonnell and Husain and in further view of "Unifying Distributed Processing and Open Hypermedia through a Heterogeneous Communication Model" by Goose et al. (hereinafter "Goose"). Applicant likewise submits that claims 5 and 6 are dependent claims that are dependent on independent claims applicant believes are allowable and therefore should be allowed.
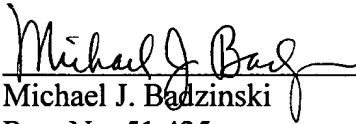
For the reasons set forth above, Applicant respectfully submits that all independent claims are believed to be in condition for allowance and that all dependent claims are believed to be dependent on allowable independent claims and therefore in condition for allowance.

Early favorable action is respectfully requested.

Please charge any additional fee occasioned by this paper to our Deposit Account
No. 03-1237.

Respectfully submitted,

Michael J. Badzinski
Reg. No. 51,425
CESARI AND MCKENNA, LLP
88 Black Falcon Avenue
Boston, MA 02210-2414
(617) 951-2500